Composite Capabilities for Cloud Manufacturing

Demonstration

Paolo Felli
Free University of Bozen-Bolzano, Italy
Paolo.Felli@unibz.it

Lavindra de Silva
University of Nottingham, UK
Lavindra.deSilva@nottingham.ac.uk

Brian Logan
University of Nottingham, UK
Brian.Logan@nottingham.ac.uk

Svetan Ratchev
University of Nottingham, UK
Svetan.Ratchev@nottingham.ac.uk

KEYWORDS

Composite capabilities; Cloud manufacturing

ACM Reference Format:
Paolo Felli, Lavindra de Silva, Brian Logan, and Svetan Ratchev. 2018. Composite Capabilities for Cloud Manufacturing. In *Proc. of the 17th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2018), Stockholm, Sweden, July 10–15, 2018,* IFAAMAS, 3 pages.

1 INTRODUCTION

We present a tool for extracting and abstracting the composite or 'collective' capabilities of a multi-agent system (MAS) given the individual capabilities of the agents in the MAS. We consider a setting where agents represent manufacturing or assembly resources such as CNC machines and robots, and the goal is to determine the composite capabilities of the manufacturing system as a whole, i.e., the products it can make. This differs from previous work that studied whether and how a particular product can be manufactured by a given set of manufacturing resources [1–3]. Our question is "*which products—or more generally, which manufacturing activities—can the agents jointly perform?*" This is key to realising the Industry 4.0 vision of flexible, adaptive and networked manufacturing systems, in which decentralised production resources form "smart factories" that communicate and collaborate [5] and where manufacturing capabilities are advertised as manufacturing services in a manufacturing cloud [6, 7].

2 MANUFACTURING RESOURCES

In our approach, each individual capability of a resource is represented as a labelled transition system (LTS), where each (possibly non-deterministic) transition is labelled with a task the agent can perform in a particular state. A task is of the form $t(\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3)$ where t is a task, and each $\mathbf{p}_i$ is a sequence of variables or constants representing parts. Additional task parameters are included in our implementation, here omitted for brevity. A resource executing t 'consumes' the *input* parts $\mathbf{p}_1$ and 'produces' the *output* parts $\mathbf{p}_3$, while the *external* parts $\mathbf{p}_2$ must be present in another resource (allowing multiple resources to perform operations in parallel on the same set of parts). *Observable tasks* correspond to manufacturing operations, while *internal tasks* are internal actions necessary to perform a manufacturing operation. Special *nop* transition labels denote 'idling', and *in* and *out* labels indicate the transfer of parts into and out of resources.

The composite capabilities of the resources are also modelled as LTSs, where transitions represent *meaningful* abstract joint operations by multiple resources. Our tool computes the legal interactions of individual resource capabilities and abstracts low-level details, to give abstract composite capabilities. Determining such composite capabilities is a non-trivial task that currently relies on the knowledge and experience of a human system integrator.

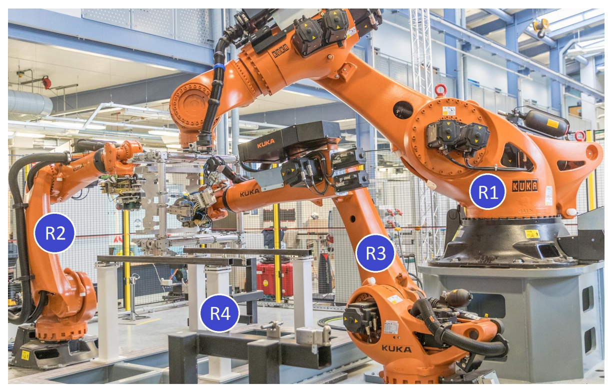

Figure 1: Our assembly resources $R_1 - R_4$.

As an example consider the flexible manufacturing cell, shown in Figure 1 and modelled in Figure 2. The cell consists of four manufacturing resources (R_1-R_4) each controlled by an agent. Resources R_1-R_3 are high-payload KUKA robots, R_4 is a shared bench equipped with clamping end effectors, and R_5 (not shown in Figure 1) is a human operator who receives instructions or alerts when a production decision must be made through an interface. Although some functionality and sensing abilities have been omitted, this cell exhibits a wide range of composite capabilities.

We briefly describe each resource and relate it to the corresponding LTS description in Figure 2. Resource R_1 is a robotic arm that can equip three end-effectors: a gripper (*eqp_g*), a drill (*eqp_d*) and a rivet gun (*eqp_r*). The latter introduces some uncontrollability in the resource, as there is no mechanism to ensure that the gun is always loaded with rivets. Similarly, R_2 can equip two types of end-effector to apply pressure against a part while another resource is performing machining operations on the part. One end-effector is hollow, to be used when the other resource is drilling, and the other is flat for applying pressure. Resource R_3 has only a gripping

Proc. of the 17th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2018), M. Dastani, G. Sukthankar, E. André, S. Koenig (eds.), July 10–15, 2018, Stockholm, Sweden.

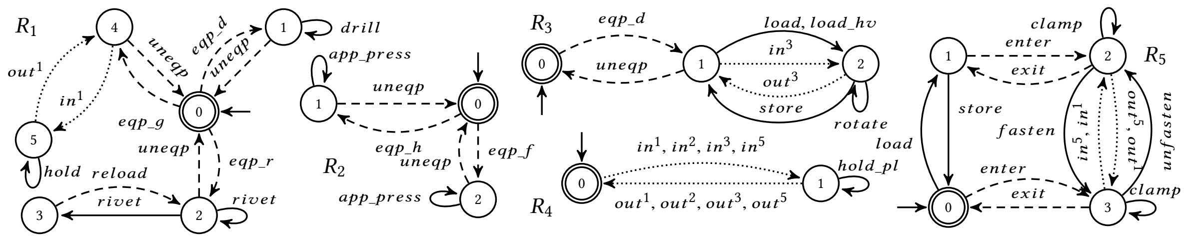

Figure 2: Models of the resources ($R_1 - R_4$) and human (R_5), with part-variables in tasks omitted.

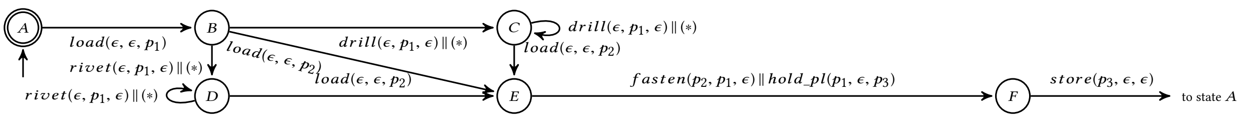

Figure 3: An extracted composite capability, where $(*)$ stands for $app\_press(\epsilon, p_1, \epsilon) \parallel hold\_pl(p_1, \epsilon, p_1)$.

end-effector that can be used to load new parts (regular or heavy) into the cell from the rack, or to rotate or store parts. Resource R_4 actively holds a part in place while the rest of the resources perform tasks on it. Finally, the operator (R_5) can enter the cell and either fasten a newly loaded part onto one that is currently held by another resource, or 'transfer' it to a robot, after which the operator can perform a clamping task. Double circles in Figure 2 represent *final states*, i.e., states in which the resource can safely be halted. Transitions labelled with observable tasks are depicted with solid lines, internal tasks with dashed lines, and transfers with dotted lines. Self-loops labelled with *nop* are available in each state except state 2 of R_5: the operator cannot be instructed to stand in the cell while holding a part indefinitely.

Together, the resources form the *production topology* representing the layout and operation of the manufacturing cell. The topology is computed by taking the cross product of the resource LTSs [2], and removing transitions whose label (set of tasks) is not 'well formed', e.g., where the external (part-)variables of a task in the set do not appear as input variables of some other task in the set. For example, in the topology representing the manufacturing cell shown in Figure 2, the transition label ($rivet, app\_press, nop, hold\_pl, nop$) denotes a joint task by the cell, where *rivet* is performed by R_1, operation *app_press* is performed in parallel by R_2, and so on. Similarly, the label ($nop, nop, out^3, in^3, nop$) represents the transfer of a part from R_3 to R_4 (while the other resources idle).

3 COMPUTING COMPOSITE CAPABILITIES

We first extract the *observable behaviour* of the resulting topology, representing the executions of the resources that are meaningful from a manufacturing perspective. To do this, we *(i)* remove 'unobservable' topology transitions by 'collapsing' into a single transition any sequence consisting of *nop*, internal, or transfer tasks and ending in an observable task; and *(ii)* create a belief-state representation [4] of the topology by merging nondeterministic transitions that cannot be controlled. This gives a topology in which each transition represents a set of joint observable tasks by all resources.

Next, each transition in the observable behaviour is replaced by one or more transitions labelled with a *task expression*, that is defined by the grammar $T := t(\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3) \mid T;T \mid T \parallel T \mid T$ '|' T, where $t(\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3)$ is a ground observable task, ';' denotes a sequence, '$\parallel$' denotes parallel composition, and '|' denotes interleaved composition [3]. For example, the joint tasks of holding and applying pressure to a part that is being drilled by a third resource can be specified as $T = drill(\epsilon, p, \epsilon) \parallel app\_press(\epsilon, p, \epsilon) \parallel hold\_pl(p, \epsilon, p)$, where ϵ is the empty string.

The final step is to extract the 'fragments' of the observable behaviour of the topology. Each fragment represents a set of possible joint executions of the resources that end in a final state. The set of fragments is then the set of composite capabilities of the topology.

As an example, Figure 3 shows a composite capability extracted by our tool corresponding to a fragment of the topology consisting of resources $R_1 - R_5$. State D represents two topology states that were merged in order to remove inherent uncontrollable nondeterminism (due to the 'rivet' transitions from state 2 in R_1), and transition $(A, load(\epsilon, \epsilon, p_1), B)$ was obtained by 'collapsing' the two topology transitions associated with transitions $(0, eqp\_d, 1)$ and $(1, load, 2)$ in R_3. Note that, while we remove uncontrollable nondeterminism, a composite capability may still have 'controllable' nondeterminism. In such cases, we assume that the alternative chosen depends on runtime tests performed during production.

The tool implements the extraction procedure for deterministic resources, and it also supports—via user-supplied rules—the abstraction of a composite capability by replacing compositions of 'facility specific' tasks appearing in one or more transitions with a single 'abstract task expression'. For example, the expression $drill(\epsilon, p, \epsilon) \parallel app\_press(\epsilon, p, \epsilon) \parallel hold\_pl(p, \epsilon, p)$ above can be abstracted to the capability $make\_hole(p, p)$ (with no external parts). This allows facility specific tasks to be translated into abstract tasks shared across a manufacturing cloud.

4 CONCLUSIONS

We have described a tool for the offline extraction and abstraction of the composite capabilities of a manufacturing system, given the individual capabilities of the available resource agents. Composite capabilities can be advertised as services in a manufacturing cloud. Once such a service is requested by a user, our tool can, similarly to [2], synthesise a controller that is able to orchestrate the resource agents in order to realise the composite capability.

REFERENCES

[1] Lavindra de Silva, Paolo Felli, Jack C. Chaplin, Brian Logan, David Sanderson, and Svetan Ratchev. 2016. Realisability of Production Recipes. In *Proceedings of the European Conference on Artificial Intelligence (ECAI)*. 1449–1457.

[2] Lavindra de Silva, Paolo Felli, Jack C. Chaplin, Brian Logan, David Sanderson, and Svetan Ratchev. 2017. Synthesising Industry-Standard Manufacturing Process Controllers. In *Proceedings of the International Conference on Autonomous Agents and MultiAgent Systems (AAMAS)*. 1811–1813.

[3] Paolo Felli, Lavindra de Silva, Brian Logan, and Svetan Ratchev. 2017. Process Plan Controllers for Non-Deterministic Manufacturing Systems. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*. 1023–1030.

[4] Leslie Pack Kaelbling, Michael L. Littman, and Anthony R. Cassandra. 1998. Planning and Acting in Partially Observable Stochastic Domains. *Artificial Intelligence* 101, 1-2 (1998), 99–134.

[5] H. Kagermann, W. Wolfgang, and J. Helbi. 2013. Recommendations for Implementing the Strategic Initiative INDUSTRIE 4.0. [Online]. Available: www.plattform-i40. de/finalreport2013. (2013). (accessed Nov 2016).

[6] Dazhong Wu, Matthew John Greer, David W. Rosen, and Dirk Schaefer. 2013. Cloud Manufacturing: Strategic Vision and State-of-the-Art. *Journal of Manufacturing Systems* 32, 4 (2013), 564 – 579.

[7] Xun Xu. 2012. From Cloud Computing to Cloud Manufacturing. *Robotics and Computer-Integrated Manufacturing* 28, 1 (2012), 75 – 86.